



© David Deen

Day 5: Looping, visualization & Assessment

What you have learned so far in week 1 ...

- Connecting to a compute cluster (AWS or BioFrontiers Fiji)
- Working on a Unix command terminal
- Running scripts processing FASTQ sequencing files
 - Quality control (FASTQC)
 - Trimming (Trimmomatic)
 - Mapping (HISAT2) to yield SAM/BAM files

What you have learned so far in week 1 ...

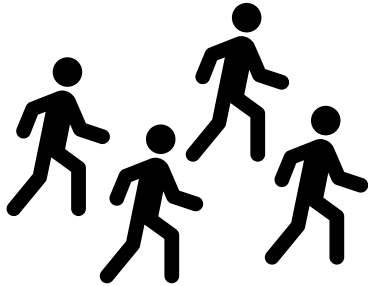
- Connecting to a compute cluster (AWS or BioFrontiers Fiji)
- Working on a Unix command terminal
- Running scripts processing FASTQ sequencing files
 - Quality control (FASTQC)
 - Trimming (Trimmomatic)
 - Mapping (HISAT2) to yield SAM/BAM files
- Goals for today:
 - Part 1:
 - `for` loops
 - Transforming mapped files to smaller visualization files
 - Part 2: Assessment of skills learned up through today

`for` loop demonstration

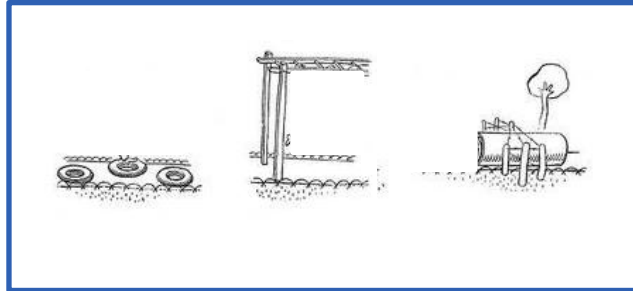
FOR_loop_basics.md worksheet under Day 5 worksheets

- This worksheet is for your reference if you need it, though you can follow along right now if you want to
- Goes over serial and parallel `for` loops with SLURM

Contestants to go through code (obstacle course). There are multiple.



The code (obstacle course) that stays the same and we want to reuse.



CPU

Fastq

Fastq

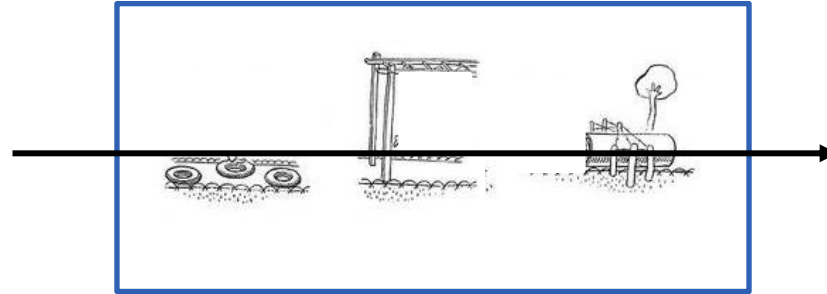
Fastq

Fastq

Script:
processing.sbatch

Contestants to go through code (obstacle course). There are multiple.

The code (obstacle course) that stays the same and we want to reuse.



CPU

Fastq

Fastq

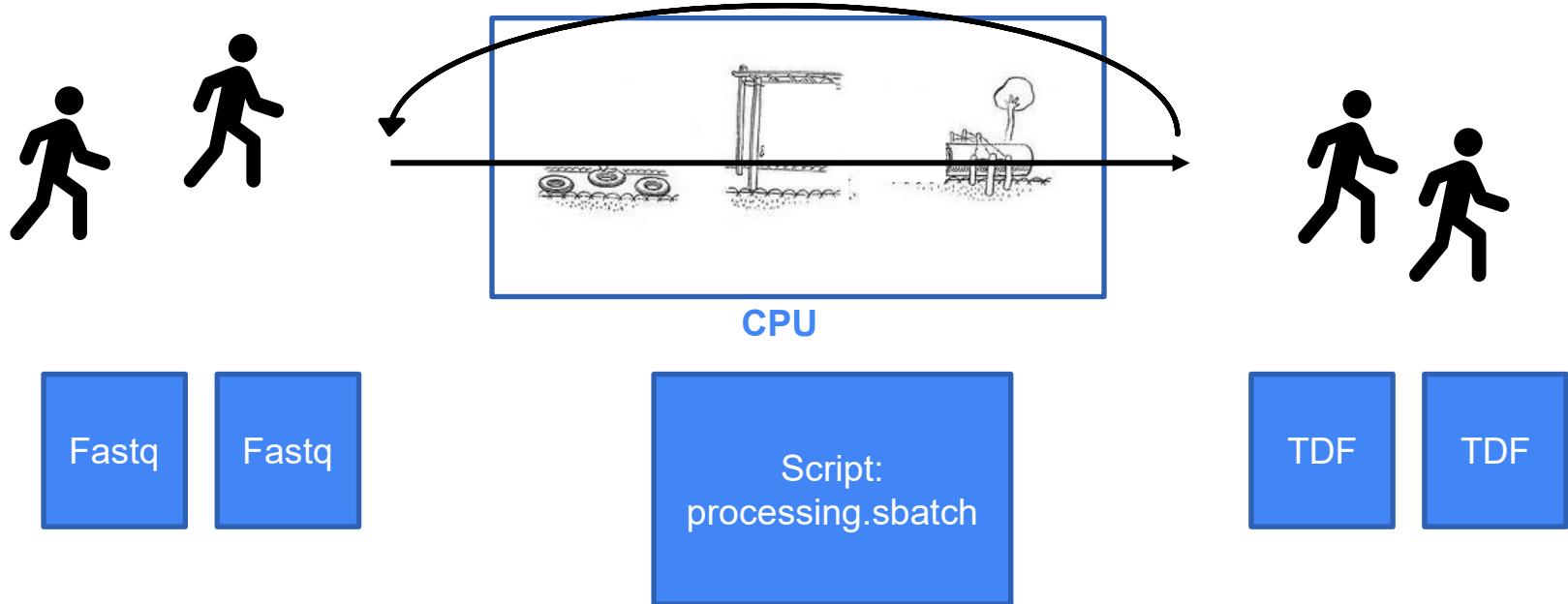
Fastq

Script:
processing.sbatch

TDF

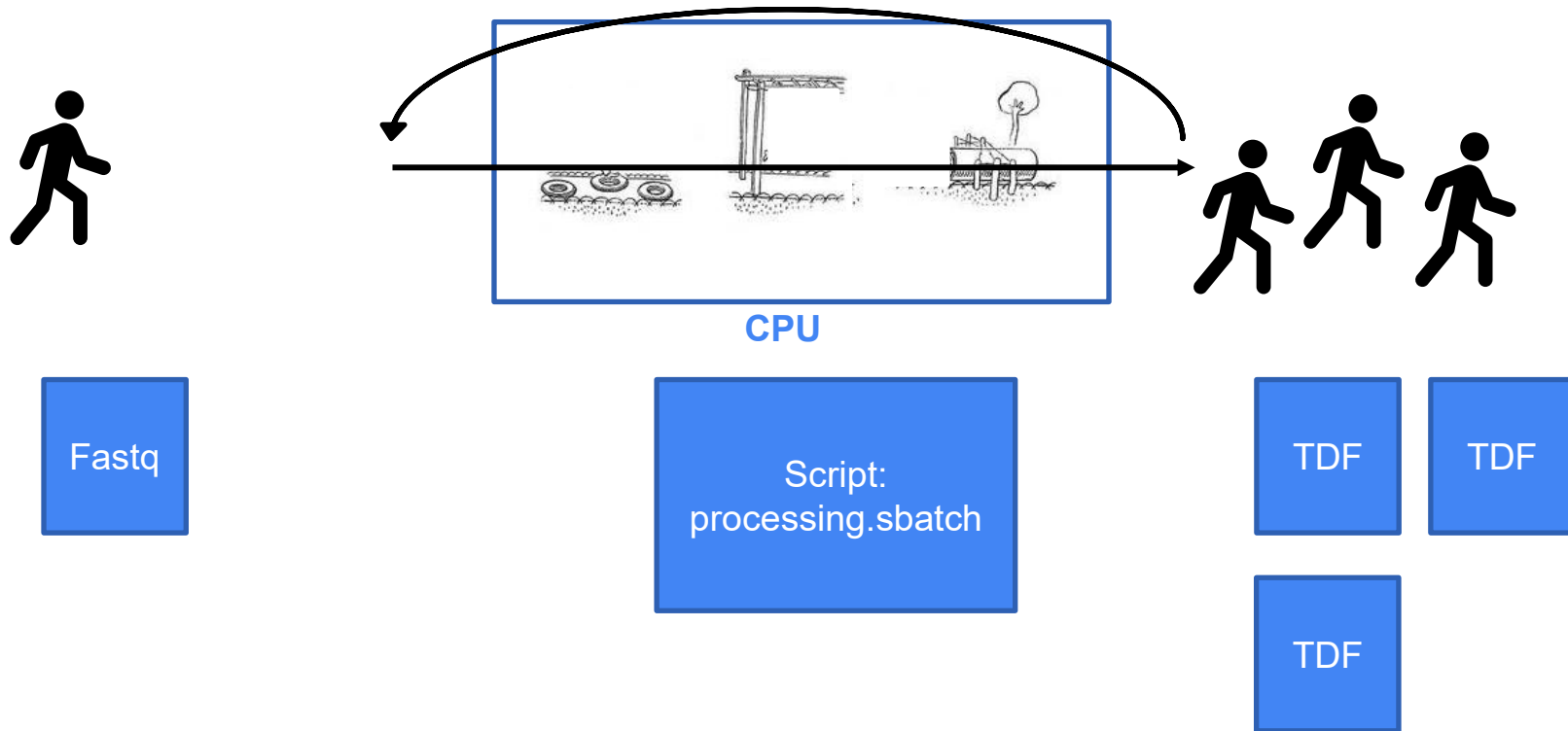
Contestants to go through code (obstacle course). There are multiple.

The code (obstacle course) that stays the same and we want to reuse.



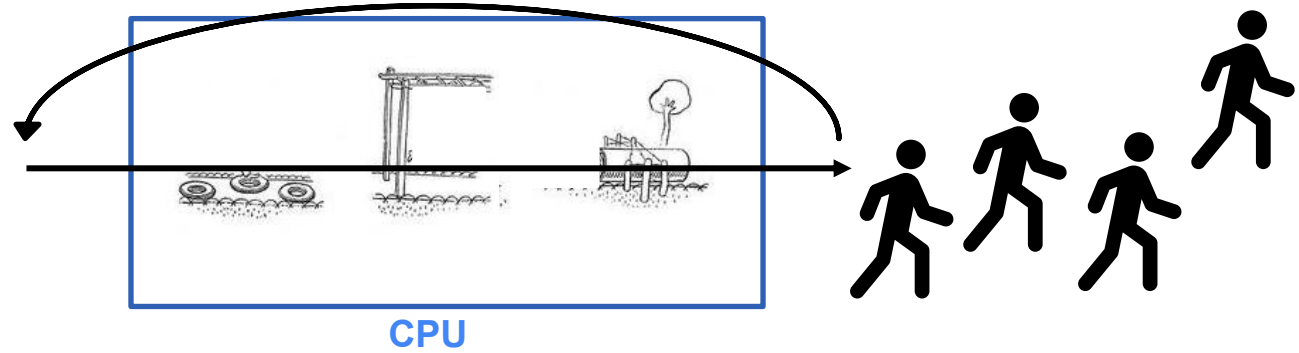
Contestants to go through code (obstacle course). There are multiple.

The code (obstacle course) that stays the same and we want to reuse.



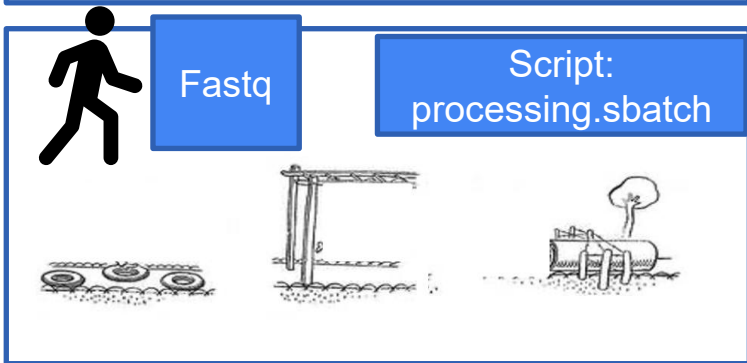
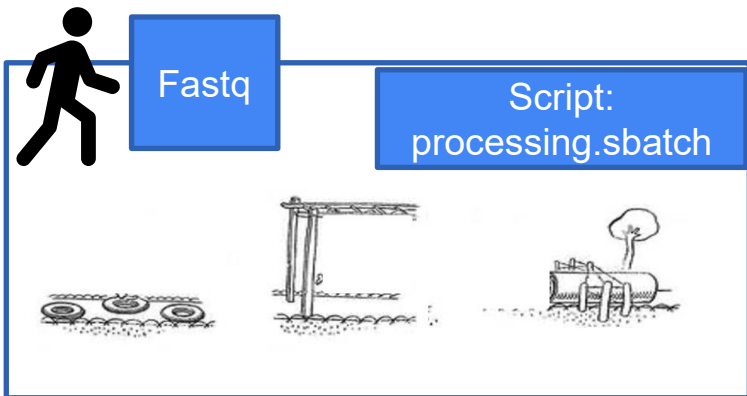
Contestants to go through code (obstacle course). There are multiple.

The code (obstacle course) that stays the same and we want to reuse.

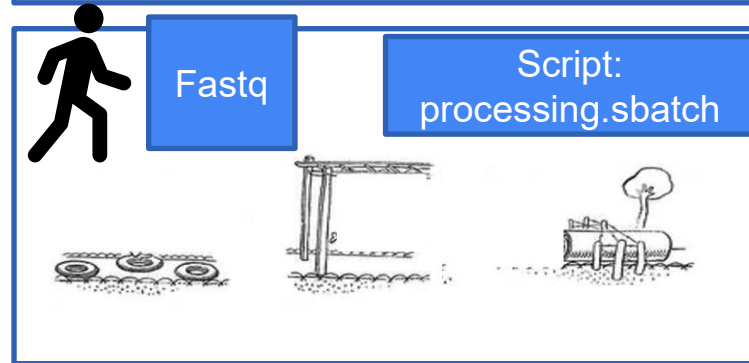
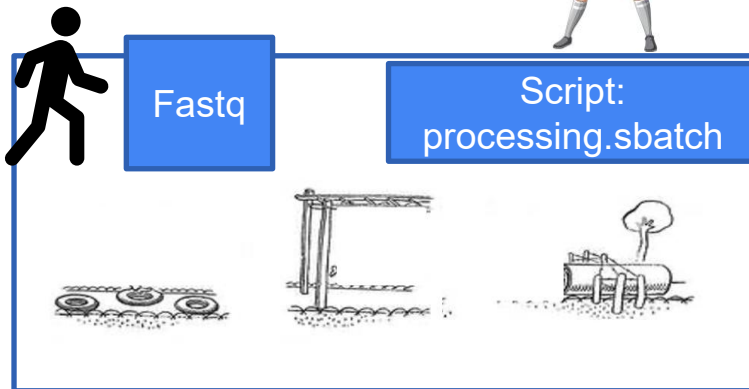


Script:
processing.sbatch

TDF	TDF
TDF	TDF

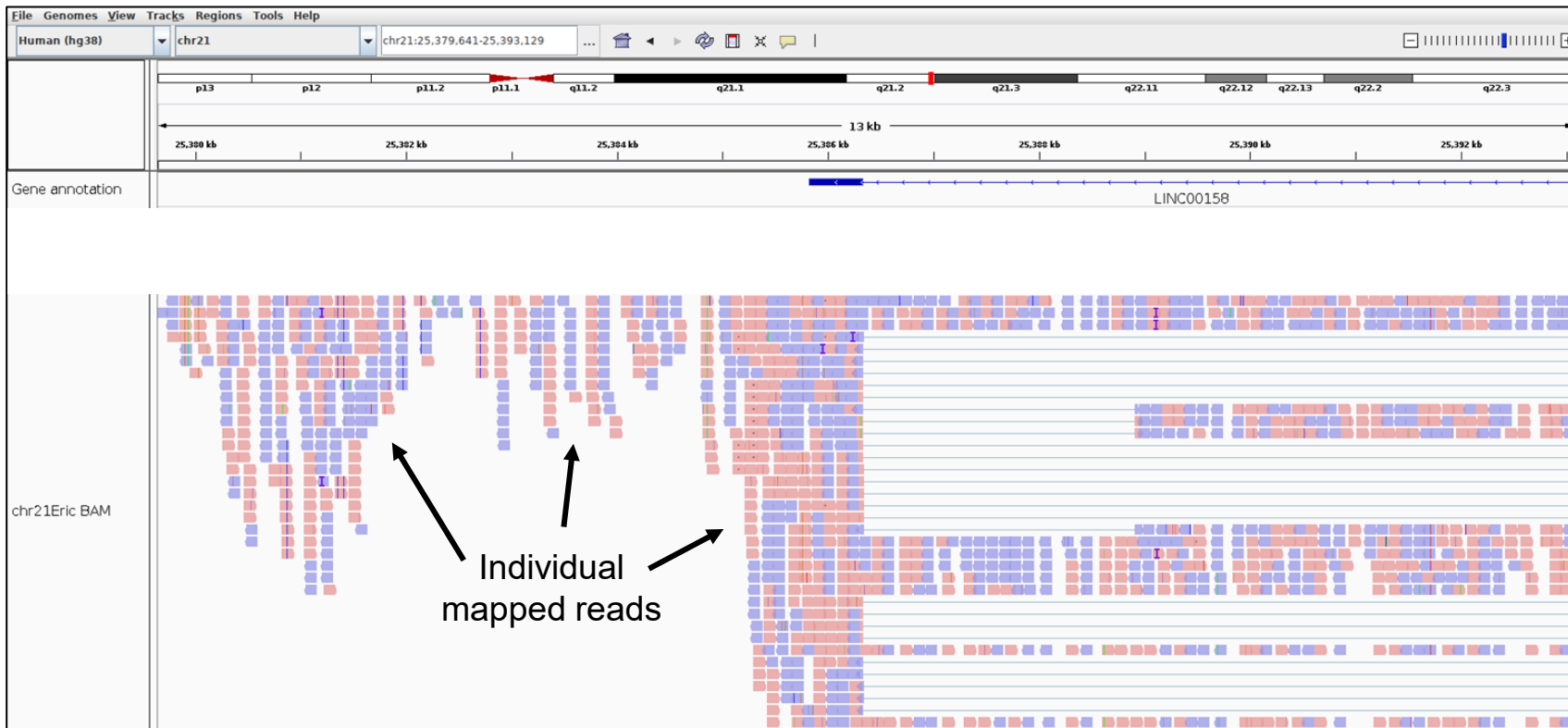


Script:
run_processing.sh



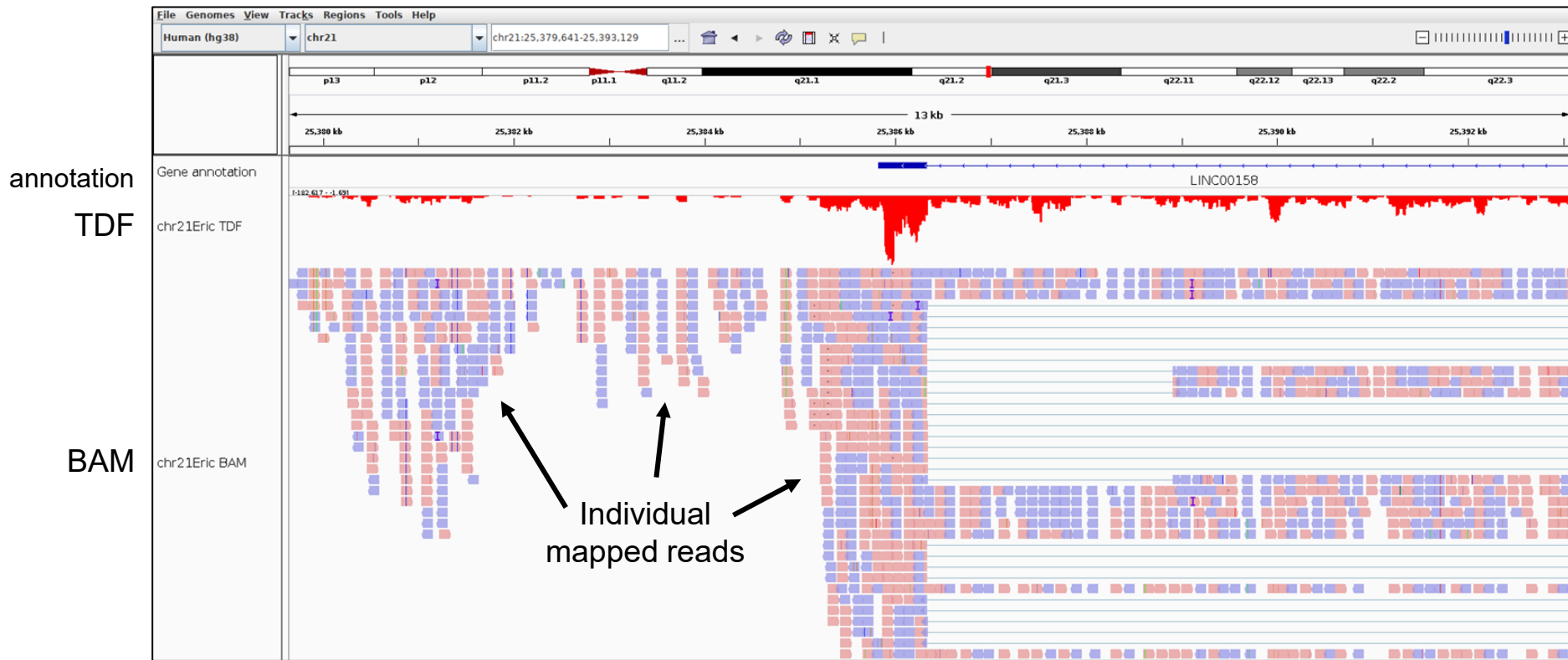
Comparing BAM and TDF files on IGV

Observed region on screen = **13 kb**



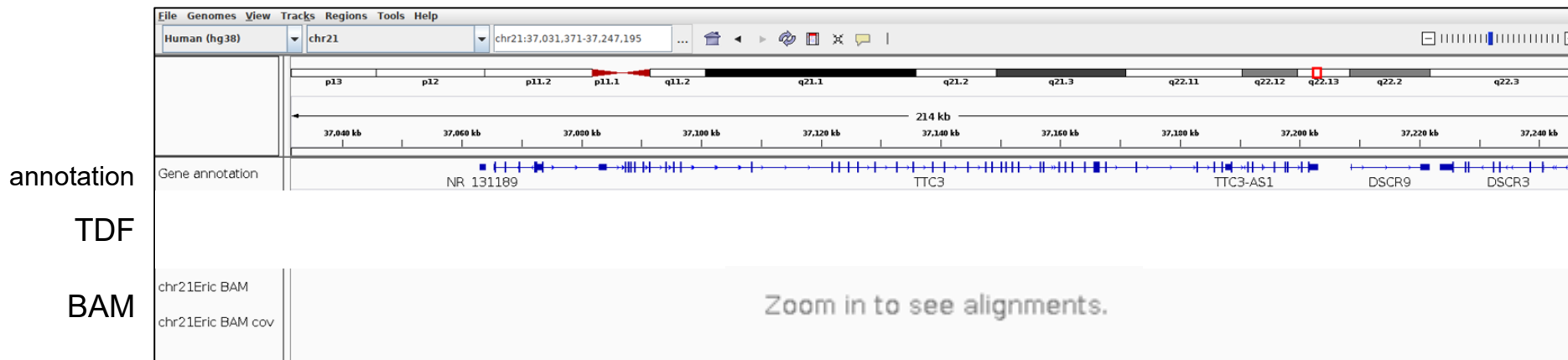
Comparing BAM and TDF files on IGV

Observed region on screen = **13 kb**



Comparing BAM and TDF files on IGV

Observed region on screen = **214 kb**

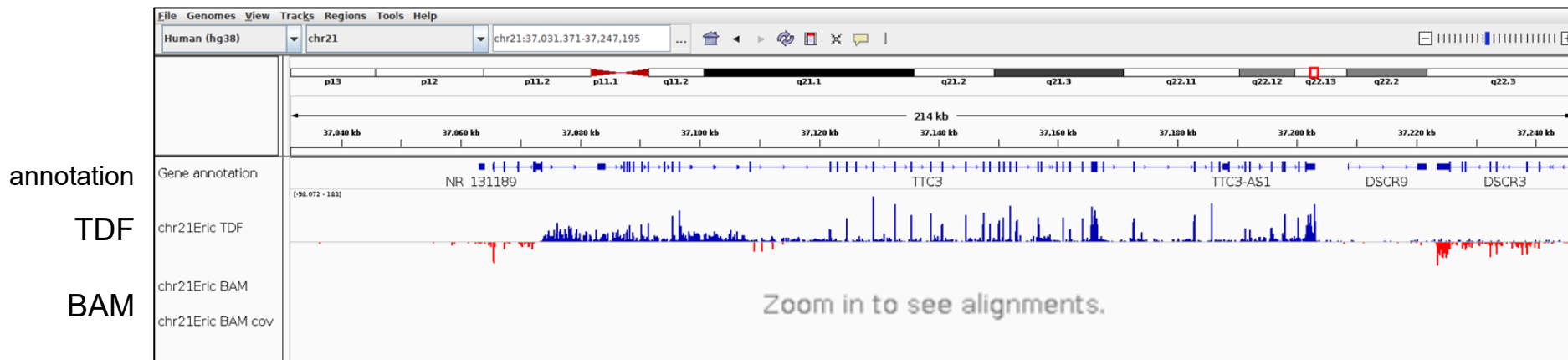


IGV does not display individually mapped reads on such a big region at once!

But IGV does okay displaying TDF coverage across any zoom region.

Comparing BAM and TDF files on IGV

Observed region on screen = **214 kb**



IGV does not display individually mapped reads on such a big region at once!

But IGV does okay displaying TDF coverage across any zoom region.

Go to the Day5 worksheet on Github,
and get started on today's exercise

First things to check when error

1. Did you check what the error and output files said?
 1. If you can't find them, check that you fixed the path in the script for these files
2. Reading an error:
 1. Usually the most helpful part of the error is at the bottom of it
3. TYPOS (especially filenames and folders)

```
-header Print the header from the A file prior to results.

-nobuf Disable buffered output. Using this option will cause each
of output to be printed as it is generated, rather than save
in a buffer. This will make printing large output files
noticeably slower, but can be useful in conjunction with
other software tools and scripts that need to process one
line of bedtools output at a time.

-iobuf Specify amount of memory to use for input buffer.
Takes an integer argument. Optional suffixes K/M/G supported.
Note: currently has no effect with compressed files.

Default Output:
After each entry in A, reports:
1) The number of features in B that overlapped the A interval.
2) The number of bases in A that had non-zero coverage.
3) The length of the entry in A.
4) The fraction of bases in A that had non-zero coverage.

***** ERROR: No input file given. Exiting. *****
```

Runtime Error:

Traceback (most recent call last):

File "", line 2, in
print(mylist[10])

IndexError: list index out of range

Day 5: Assessment

Check out the other FASTQ datasets in Day5.

Imagine these files are fresh off the sequencer... can you write your own scripts to turn them from raw FASTQs to TDFs? What QC checks should you perform?

(Extra-very-real-points: Do all files with a loop!)



FASTQ → SAM → BAM →
→ BEDGRAPH → TDF



First things to check when error

1. Did you check what the error and output files said?
 1. If you can't find them, check that you fixed the path in the script for these files
2. Reading an error:
 1. Usually the most helpful part of the error is at the bottom of it
3. TYPOS (especially filenames and folders)

```
-header Print the header from the A file prior to results.

-nobuf Disable buffered output. Using this option will cause each
of output to be printed as it is generated, rather than save
in a buffer. This will make printing large output files
noticeably slower, but can be useful in conjunction with
other software tools and scripts that need to process one
line of bedtools output at a time.

-iobuf Specify amount of memory to use for input buffer.
Takes an integer argument. Optional suffixes K/M/G supported.
Note: currently has no effect with compressed files.

Default Output:
After each entry in A, reports:
1) The number of features in B that overlapped the A interval.
2) The number of bases in A that had non-zero coverage.
3) The length of the entry in A.
4) The fraction of bases in A that had non-zero coverage.

***** ERROR: No input file given. Exiting. *****
```

Runtime Error:

Traceback (most recent call last):

File "", line 2, in
print(mylist[10])

IndexError: list index out of range